# CS 560: Computer Graphics

## Assignment 3 Report

**Contact Information:**

Name: Ved Ranade

Email ID: vranade1@binghamton.edu

## 1. Problem Statement:

The purpose of completing this assignment is to learn 3D camera operations and applying object transformation in 3D space using OpenGL. The assignment consists of the following tasks:

- Implement 4 different viewport in four corners of the screen.
- In the first viewport, draw a ground and a cylinder
- Set the camera at a particular point and perform the rolling, pitching and yawing functions on the camera by changing parameters of the OpenGL glulookat() function
- Also move the camera along the Y and Z axis
- In the other viewports, draw the specified level object
- In viewport V2, place the camera on top of the object looking down on it (top view)
- In viewport V3, place the camera on the side of the object looking towards it (side view)
- In viewport V4, place the camera in front of the object looking towards it (front view)
- Rotate the object as specified by applying 3D matrix transformations.

## 2. Algorithm design:

Viewport V1:

- Set the perspective mode
- Draw a ground and a cylinder
- Set a camera using glulookat() pointing to the cylinder

Operations in V1:

- To roll left, decrease the third last parameter in glulookat(). To roll right, increase the third last parameter in glulookat()
- For yawing, increase or decrease the $4^{th}$ parameter in gluLookAt()
- For pitching, increase or decrease the $5^{th}$ parameter in gluLookAt()
- For moving forward or backwards, increase or decrease the 3rd parameter in gluLookAt()
- For moving sideways or up/down, modify the $1^{st}$ or $2^{nd}$ parameters in gluLookAt()

Setting V2, V3, V4:

- Set the required camera angles by changing the parameters of gluLookAt()
- To draw the required object, first draw the object at origin by calling the OpenGL primitives
- Then move each individual object in its position by applying glTranslatef()

Operations in V2, V3, V4:

- For each individual object in the structure, transport the object to origin using glTranslatef()
- Apply the required rotation using glRotatef()
- Move the object back to its appropriate place in the structure using glTranslatef()

## 3. Running the program:

***OpenGL and GLUT were used for the execution of this program***

Step 1: Unzip the uploaded files to a location

Step 2: Find and run the file named CGHomework3.exe in the extracted directory. You may also compile and execute the main.cpp file if you are running the program in Linux

Step 3: The application window will come up and a menu will be displayed. Select the appropriate choice

Step 4: Controls:

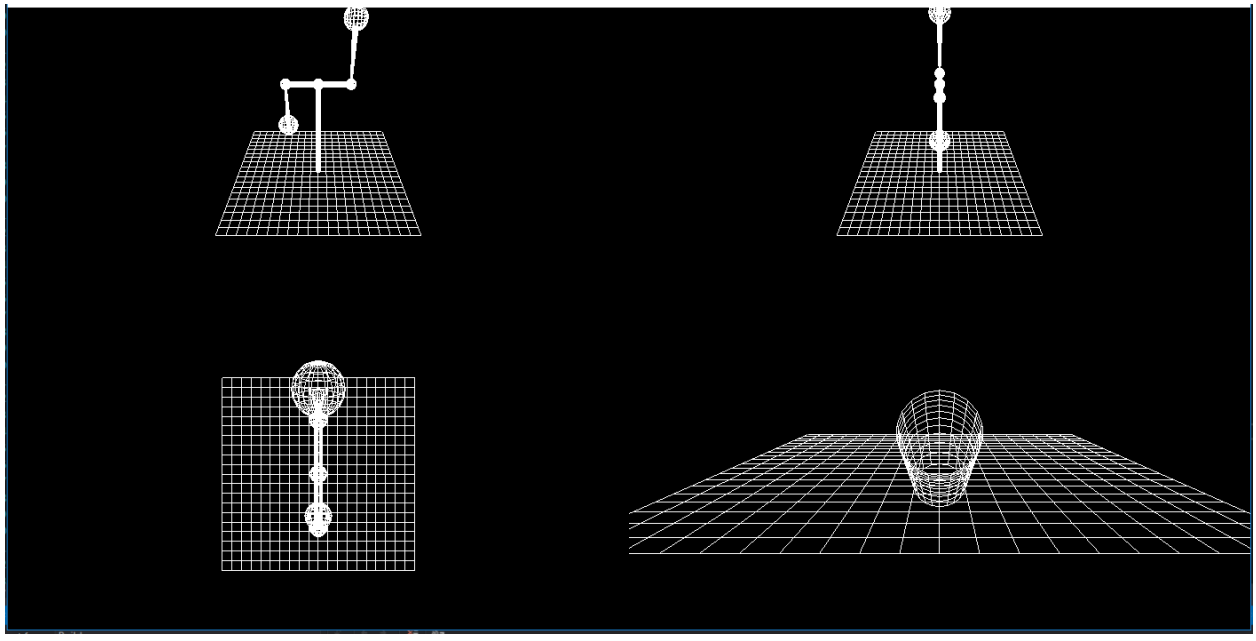
W = move forward, s = move backward, q = yaw left, e = yaw right, a = shift left, d = shift right,

Z = roll left, x = roll right, f = pitch up, v = pitch down,

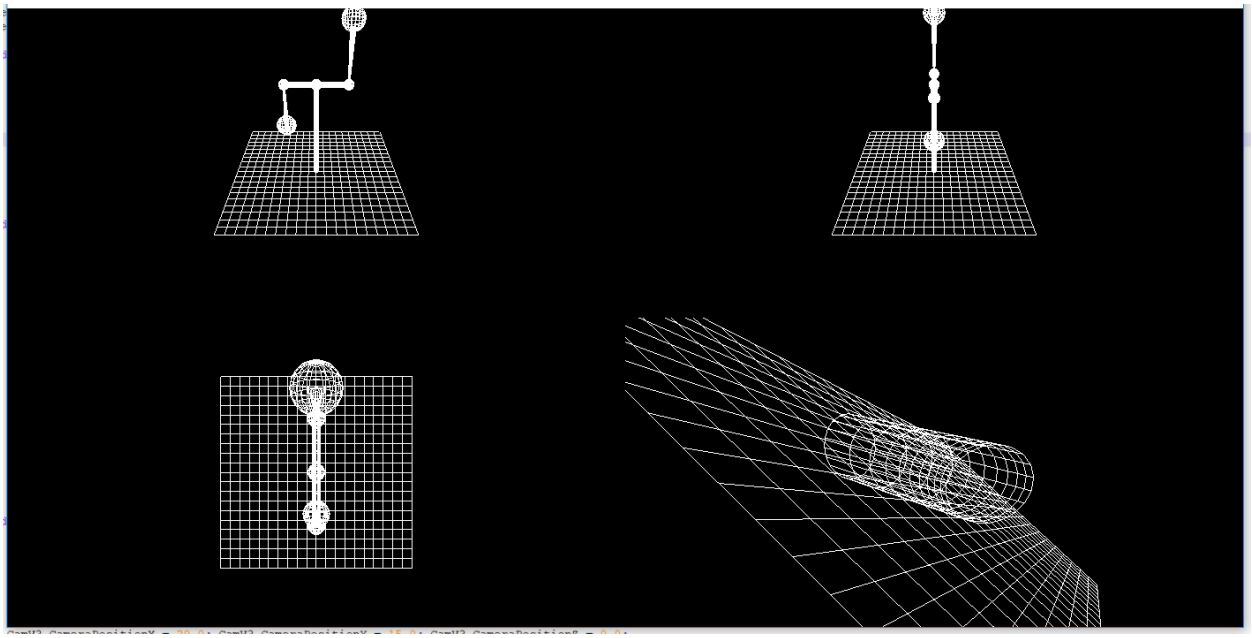
t = rotate object in V2, V3 and V4

## 4. Results

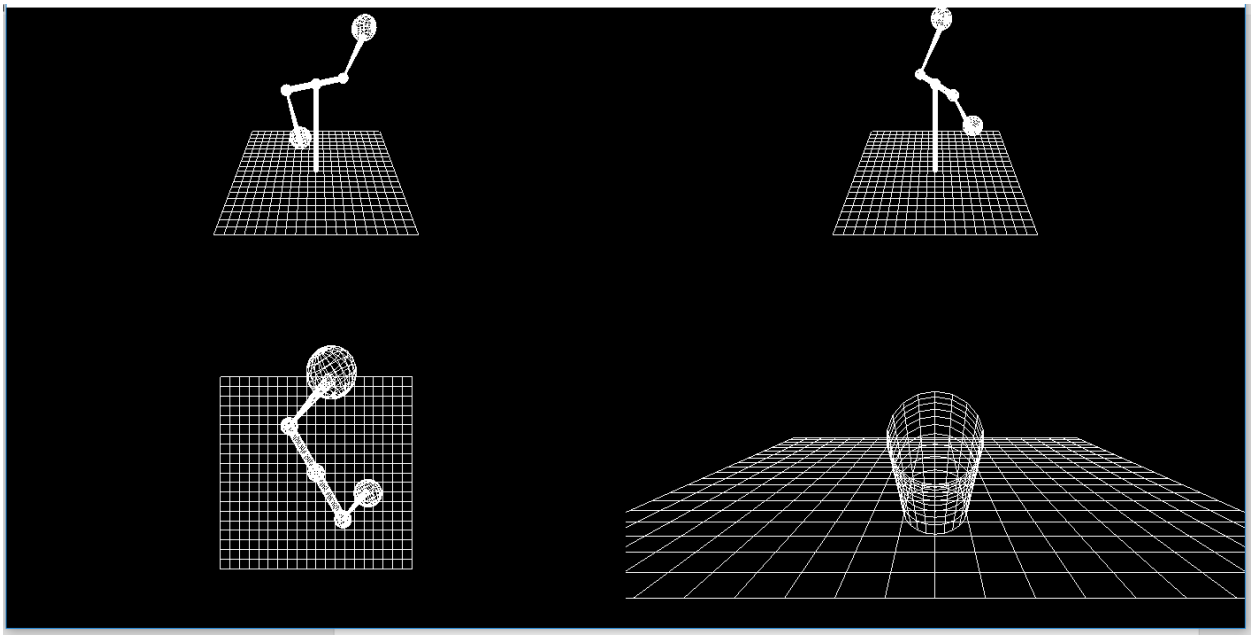
Original setup:

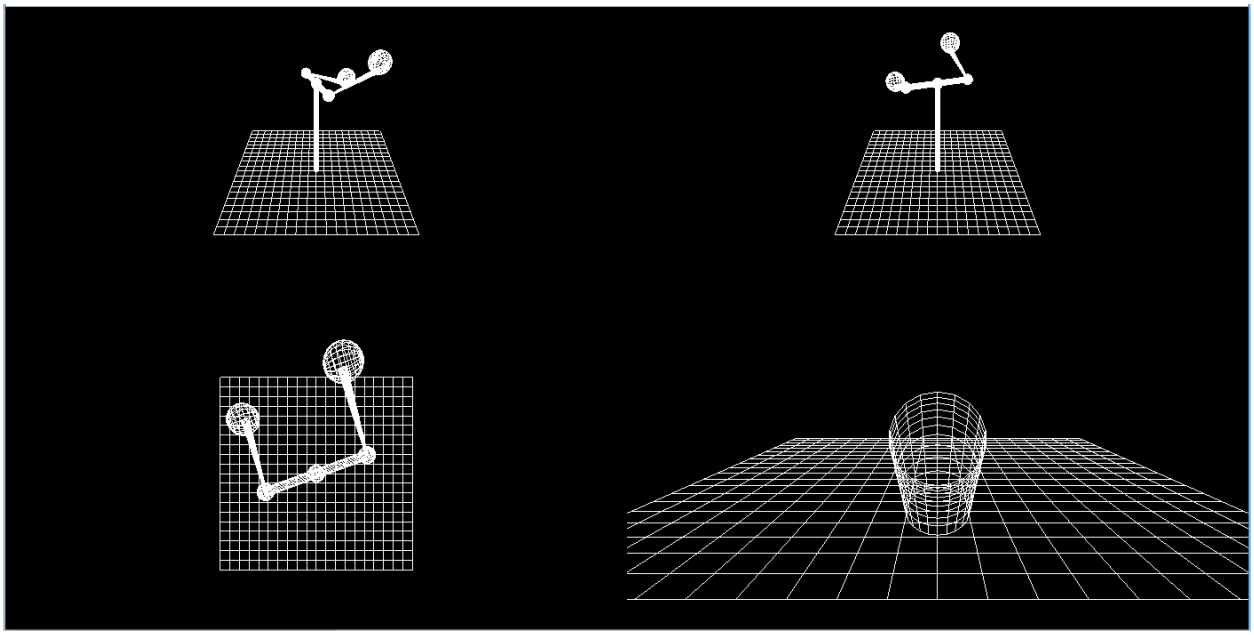


Yawing and rolling in V1:
Using keys ‘q’ and ‘z’



Moving the object in V2, V3 and V4, using key ‘t’:





## 5. Major codes:

Moving the camera in V1 using w, a ,s,d keys:

```
void Camera::PartAHandleKeyboard(unsigned char key, int x, int y)
{
    switch (key)
    {
        case 'w':
            CamPartA.CameraPositionZ = CamPartA.CameraPositionZ - 0.5;
            CamPartA.CameraPointingToZ = CamPartA.CameraPointingToZ - 0.5;
            CameraIsMoved = true;
            glutPostRedisplay();
            break;
        case 's':
            CamPartA.CameraPositionZ = CamPartA.CameraPositionZ + 0.5;
            CamPartA.CameraPointingToZ = CamPartA.CameraPointingToZ + 0.5;
            CameraIsMoved = true;
            glutPostRedisplay();
            break;
        case 'd':
            CamPartA.CameraPointingToX = CamPartA.CameraPointingToX + 0.5;
            CamPartA.CameraPositionX = CamPartA.CameraPositionX + 0.5;
            CameraIsMoved = true;
            strafe -= 10;
            glutPostRedisplay();
            break;
        case 'a':
            CamPartA.CameraPointingToX = CamPartA.CameraPointingToX - 0.5;
            CamPartA.CameraPositionX = CamPartA.CameraPositionX - 0.5;
            CameraIsMoved = true;
            strafe += 10;
            glutPostRedisplay();
            break;
```

Generating view V3:

```
void GenerateV3()
{
    glViewport(0, 350, 700, 350);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(60.0, 2.0, 1.0, 256.0);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    CamV3.CameraPositionX = 20.0; CamV3.CameraPositionY = 15.0; CamV3.CameraPositionZ = 0.0;
    CamV3.CameraPointingToX = 0.0; CamV3.CameraPointingToY = 0.0; CamV3.CameraPointingToY = 0.0;
    CamV3.CameraTiltX = 0.0; CamV3.CameraTiltY = 1.0; CamV3.CameraTiltZ = 0.0;
    gluLookAt
    (
        CamV3.CameraPositionX, CamV3.CameraPositionY, CamV3.CameraPositionZ,
        CamV3.CameraPointingToX, CamV3.CameraPointingToY, CamV3.CameraPointingToY,
        CamV3.CameraTiltX, CamV3.CameraTiltY, CamV3.CameraTiltZ
    );
    GenerateObject();
    DrawGround();
}
```

Generating object for V2, V3, V4:

```
void GenerateObject()                          //Generates the object for viewpoints V2, V3, V4
{
    glPushMatrix();
    glTranslatef(0.0, 7.0, 0.0);
    glRotatef(90.0, 1.0, 0.0, 0.0);                                    //Making the cylinder upright
    //glScalef(1.0, 2.0, 1.0);
    //Draw cylinder S1
    GLUquadricObj* VerticalCylinderS1 = gluNewQuadric();
    gluQuadricDrawStyle(VerticalCylinderS1, GLU_LINE);
    gluCylinder(VerticalCylinderS1, 0.2, 0.2, 8.0, 20, 10);                 //BaseRadius = 0.2, TopRadius = 0.2, Height = 8.0
    glPopMatrix();

    glPushMatrix();
    glTranslatef(0.0, 7.5, 0);
    //glScalef(1.0, 2.0, 1.0);
    //Draw sphere B2
    GLUquadricObj* SphereB2 = gluNewQuadric();
    gluQuadricDrawStyle(SphereB2, GLU_LINE);
    gluSphere(SphereB2, 0.4, 20, 10);
    glPopMatrix();

    glPushMatrix();
    glRotatef(RotationPY, 0.0, 1.0, 0.0);
    glTranslatef(0.0, 7.5, -2.5);
    //glScalef(1.0, 2.0, 1.0);
    //Draw cylinder S2 S3
    GLUquadricObj* CylinderS2S3 = gluNewQuadric();
    gluQuadricDrawStyle(CylinderS2S3, GLU_LINE);
    gluCylinder(CylinderS2S3, 0.2, 0.2, 5.0, 20, 10);
    glPopMatrix();
```

## 6. Extra credit attempted:

- Moving camera in an arbitrary direction, i.e. along Y or X axis